Relatório de Tolerância de Faltas

Grupo T59

http://www.github.com/tecnico-distsys/T59-Komparator

Ricardo Filipe
84621

Tiago Da Silva Letra
84627

O protocolo implementado no projeto permite a replicação passiva do servidor mediador e em caso de falha seja substituído por uma réplica, o protocolo também garante a semântica no cliente é no-máximo-uma-vez. O protocolo também permite um número dinâmico de réplicas.

Replicação Passiva

Para implementar a replicação passiva sempre que o cliente enviar um pedido com a operações buyCart(...) ou addToCart(...), o servidor principal ao recebe-los envia um pedido one-way updateShopHistory(ShopResultView shopResult,String idOperation) ou updateCart(CartView cart, String idOperation), consoante o pedido. As operações update são compostas por 2 argumentos, um é o objeto que foi criado com a operação (cart ou shopResult) e outro é responsável por garantir a semântica (idOperation), explicado como mais tarde no relatório.

As réplicas do servidor ao receberem as operações update, guardam o objeto e idOperation em estruturas, sincronizando-se com o servidor principal.

Um pormenor, a implementação não garante que após o servidor principal ter sido iniciado e ter feito operações que se encontre sincronizado com réplica que tenha sido criado após as operações, para garantir esta sincronização é necessário fazer um pedido ao Servidor principal com a operação clean(). A operação clean foi modificada, quando um servidor principal recebe-a, ele propaga para todos as réplicas.

Tolerância a falhas

No projeto nem todo tipo de falhas são toleradas, o protocolo implementado permite haver tolerância a falhas no servidor, pois no caso de existir uma falha no servidor ele é substituído por uma réplica e réplica assim torna-se o principal.

A réplica para saber o estado do servidor principal, é usado uma tarefa em cada servidor (réplicas e principal). A tarefa do servidor principal envia continuamente num intervalo de 5 segundos (intervalo pode ser configurado) um pedido one-way com a operação imAlive() para todas as réplicas. As réplicas ao receberem este pedido guardam a data e a hora de quando foi recebido, as réplicas apenas guardam a data e hora do último pedido. A tarefa das réplicas verifica a cada 5 segundos (mesmo intervalo que a tarefa do servidor principal), se recebeu um pedido imAlive() recente, um pedido é recente se foi recebido à menos de tempo que (intervalo de imAlive() + tempo máximo que recetor demora a receber a mensagem). Se o pedido não for recente, uma das réplicas torna-se o servidor principal.

A réplica que se torna o servidor principal é réplica com valor do porto mais próximo ao valor do porto do servidor principal, para saber isso as réplicas usam UDDI.

No projeto também existe tolerância a falhas na comunicação entre o cliente e o servidor principal, através do uso de um Front-End, sempre que um pedido sofre time-out ou connection refused, o cliente espera 2 ciclos das tarefas + o tempo máximo que pedido demora receber depois de enviado (11 segundos neste caso), depois verifica, através do UDDI, se alguma réplica se tornou o servidor principal e volta a realizar o pedido, isto repete-se um certo número de vezes até desistir (número configurável). O

cliente espera 2 ciclos das tarefas porque esse é o tempo máximo para as réplicas de aperceberem que o servidor principal falhou e atualizou o UDDI. No caso de time-out o cliente espera tempo adicional (1-2 vezes tempo máximo que o pedido demora a chegar depois de enviado, pode ser configurado depende do time-out). Em caso de connection refused, o servidor principal já falhou antes de receber a mensagem e o a réplica ainda não teve tempo de atualizar o UDDI que demora no máximo 2 ciclos da tarefa + tempo máximo que o pedido demora a chegar ao UDDI, portanto o cliente não precisa de esperar segundos adicionais.

Não existe tolerância a falhas na comunicação entre servidor principal e as réplicas, entre os servidores e o UDDI e entre os clientes e o UDDI.

Garantir a Semântica

No projeto só existem 2 operações idempotentes, entre o servidor e o cliente, o buyCart(...) e o addToCart(...), para garantir a semântica é necessário identificar cada pedido para que a operação só seja executada uma vez, pois ao receber um pedido com o mesmo identificador, não a executa apenas devolve o mesmo resultado da operação do pedido com o mesmo id. Isto é importante porque caso uma reposta se perda e o cliente volte a repetir o pedido, o servidor não volta a executar as operações idempotentes.

O identificador é inserido no pedido através do uso de handlers, o identificador é adicionado ao cabeçalho antes de ser enviado pelo cliente (IdOPerationSender) e o handler IdOPerationReceiver do servidor é responsável por retira-lo do cabeçalho e dar ao servidor.

O identificador também tem que se enviador nos updates para impedir que no caso do servidor principal falhe após de enviar uma resposta que se perdeu e a réplica que o substitui-o volte a executar a mesma operação e garantir a semântica. No caso de as réplicas receberem updates de shopResultView que já receberam, as réplicas ignoram o pedido. No caso da update de cartView, o servidor não envia update do cartView.

O identificador do pedido é formado por 2 componentes , uma é identificador do cliente (diferencia o pedido dos outros clientes) e outra é uma variável que é incrementada sempre que cliente quer envia um pedido com as operações idempotentes (diferencia o pedido dos outros pedidos). O identificador do cliente é composto pelo MAC address, hashcode da hora e por fim identificador Universal.